Recap: Physics-Informed Neural Networks Example: A 2D Poisson PDE, and Black-Scholes with 2 risky assets

Simon Scheidegger

March 27th, 2025

PINN: Concept

- **Objective**: Solve PDE by embedding the PDE residual into a neural network's loss.
- Instead of only fitting data, we also enforce:

$$\text{Loss} = \underbrace{\text{MSE}(\text{PDE residual})}_{\text{physics}} + \underbrace{\text{MSE}(\text{boundary conditions})}_{\text{boundary data}}.$$

- Use automatic differentiation (`autograd`) to compute PDE derivatives inside the network.
- Minimization via standard optimizers (e.g. Adam) yields an approximate solution.

PDE Setup

2D Poisson Problem:
We consider the domain $\Omega = [0, 1] \times [0, 1]$ with the PDE

$$\Delta u(x, y) \;=\; 4, \quad (x, y) \in \Omega,$$

and the **Dirichlet boundary condition**

$$u(x, y) \;=\; x^2 + y^2, \quad (x, y) \in \partial\Omega.$$

Analytical solution:
It is straightforward to check that

$$u^*(x, y) \;=\; x^2 + y^2$$

satisfies both the PDE

$$\Delta(x^2 + y^2) \;=\; 2 + 2 \;=\; 4$$

and the same boundary condition.

- **PDE Residual:** $\Delta u - 4 = 0$
- **Boundary Mismatch:** $u - (x^2 + y^2) = 0$

PINN Architecture

- **Network**: fully-connected, Tanh activations.
- **Inputs**: (x, y) coordinates.
- **Output**: $u_\theta(x, y)$, neural approximation.
- **Loss function**:

$$\mathcal{L}(\theta) = \frac{1}{N_{\text{int}}} \sum_{\text{int pts}} \left[\Delta u_\theta - 4\right]^2 + \frac{1}{N_{\text{bc}}} \sum_{\text{bc pts}} \left[u_\theta - (x^2 + y^2)\right]^2.$$

Results & Accuracy

- We train with random points in the domain for PDE, and on edges for boundary conditions.
- **Metrics**: MSE of PDE residual (interior) and boundary mismatch.
- After training, we evaluate:

$$\mathrm{MSE}_{\mathrm{interior}} = \frac{1}{|\Omega_{\mathrm{test}}|} \sum_{\Omega_{\mathrm{test}}} \left[u_\theta - u^*\right]^2,$$

$$\mathrm{MSE}_{\mathrm{boundary}} = \frac{1}{|\partial\Omega_{\mathrm{test}}|} \sum_{\partial\Omega_{\mathrm{test}}} \left[u_\theta - u^*\right]^2.$$

- Typically, errors can reach 10^{-5} to 10^{-7} range, depending on network size and training steps.

Basket Options: An Overview

- **Basket Options** are derivatives written on a weighted average of multiple assets.
- Consider a European call option on a basket:

$$\text{Payoff} = \max\Big(w_1 S_1 + w_2 S_2 - K,\, 0\Big)$$

- In our example, we choose:

$$w_1 = w_2 = 0.5, \quad K = 100, \quad T = 1.$$

- The assets have different volatilities and a nonzero correlation.

Dynamics of Underlying Assets

Assume under the risk-neutral measure the asset prices follow Geometric Brownian Motion:

$$dS_1 = rS_1\,dt + \sigma_1 S_1\,dW_1,$$

$$dS_2 = rS_2\,dt + \sigma_2 S_2\,dW_2,$$

with

$$dW_1\,dW_2 = \rho\,dt.$$

- r: Risk-free rate.
- σ_1, σ_2: Volatilities.
- ρ: Correlation.

Derivation of the PDE (I)

Using Itô's formula for a twice-differentiable function $V(t, S_1, S_2)$, we have

$$dV = V_t\,dt + V_{S_1}\,dS_1 + V_{S_2}\,dS_2 + \frac{1}{2}V_{S_1S_1}(dS_1)^2 + \frac{1}{2}V_{S_2S_2}(dS_2)^2 + V_{S_1S_2}\,dS_1\,dS_2.$$

Substitute the dynamics of S_1 and S_2:

$$\begin{aligned} dV = {} & V_t\,dt + rS_1V_{S_1}\,dt + rS_2V_{S_2}\,dt \\ & + \frac{1}{2}\sigma_1^2S_1^2V_{S_1S_1}\,dt + \frac{1}{2}\sigma_2^2S_2^2V_{S_2S_2}\,dt \\ & + \rho\sigma_1\sigma_2S_1S_2V_{S_1S_2}\,dt + \text{stochastic terms.} \end{aligned}$$

In a risk-neutral world, the discounted price $e^{-rt}V(t, S_1, S_2)$ must be a martingale. Thus, the drift must vanish:

$$V_t + rS_1V_{S_1} + rS_2V_{S_2} + \frac{1}{2}\sigma_1^2S_1^2V_{S_1S_1} + \frac{1}{2}\sigma_2^2S_2^2V_{S_2S_2} + \rho\sigma_1\sigma_2S_1S_2V_{S_1S_2} - rV = 0.$$

Derivation of the PDE (II)

- The PDE is solved backwards in time, i.e., t represents time-to-maturity.
- **Terminal condition:**

$$V(T, S_1, S_2) = \max\Big(w_1 S_1 + w_2 S_2 - K,\ 0\Big).$$

- For our example, with $w_1 = w_2 = 0.5$ and $K = 100$, the terminal payoff is:

$$V(T, S_1, S_2) = \max\Big(0.5 S_1 + 0.5 S_2 - 100,\ 0\Big).$$

- To make the computational domain finite, we truncate $S_1, S_2 \in [0, S_{\max}]$ (with, say, $S_{\max} = 200$) and impose homogeneous Neumann conditions (zero flux) at the boundaries.

Physics-Informed Neural Networks (PINNs)

- **Idea:** Approximate the solution $V(t, S_1, S_2)$ with a neural network $V_\theta(t, S_1, S_2)$.
- **Loss Function:**

$$\mathcal{L}(\theta) = \underbrace{\frac{1}{N_{\text{int}}} \sum_{i=1}^{N_{\text{int}}} \Big| \mathcal{R}(t_i, S_{1,i}, S_{2,i}; \theta) \Big|^2}_{\text{Interior (PDE) Loss}}$$

$$+ \underbrace{\frac{1}{N_{\text{term}}} \sum_{j=1}^{N_{\text{term}}} \Big| V_\theta(T, S_{1,j}, S_{2,j}) - \text{Payoff}_j \Big|^2}_{\text{Terminal Loss}}$$

$$+ \underbrace{\mathcal{L}_{\text{neum}}}_{\text{Neumann Loss}} \quad .$$

- **PDE Residual:** $\mathcal{R} = V_t + \frac{1}{2}\sigma_1^2 S_1^2 V_{S_1 S_1} + \frac{1}{2}\sigma_2^2 S_2^2 V_{S_2 S_2} + \rho\sigma_1\sigma_2 S_1 S_2 V_{S_1 S_2} + rS_1 V_{S_1} + rS_2 V_{S_2} - rV$.
- The training minimizes $\mathcal{L}(\theta)$ using gradient-based optimization.

PINN Training Process

- **Step 1:** Define the neural network architecture $V_\theta(t, S_1, S_2)$.
- **Step 2:** Use automatic differentiation to compute the derivatives in the PDE residual.
- **Step 3:** Sample points from the interior and boundaries:
  - Interior points for the PDE residual.
  - Terminal points (at $t = T$) for the payoff.
  - Spatial boundary points for enforcing Neumann conditions.
- **Step 4:** Construct the total loss as the sum of the interior, terminal, and Neumann losses.
- **Step 5:** Train the network by minimizing the total loss with an optimizer (e.g., Adam).

Implementation Details: Code Overview

- **Network:** The class `PINN_BS` implements a feed-forward network with three inputs (t, S_1, S_2) and one output V.
- **PDE Residual:** Function `bs_pde_residual` computes the residual using autograd.
- **Loss Functions:**
  - `interior_loss` samples interior points and evaluates the PDE residual loss.
  - `terminal_loss` enforces the terminal condition.
  - `neumann_loss` enforces homogeneous Neumann conditions on the spatial boundaries.
- **Training:** The function `train_pinn_bs` combines these losses and trains the model.
- **Visualization:** 3D scatter plots are generated at $t = T$ to compare the exact terminal solution, the PINN prediction, and the absolute error.

Results: Training Curves and Metrics

- During training, the interior loss, terminal loss, and Neumann loss are tracked.
- At convergence, we report:
  - $\text{MSE}_{\text{interior}}$ on the domain.
  - $\text{MSE}_{\text{terminal}}$ at $t = T$.
  - Neumann error on the spatial boundaries.
- These metrics indicate the overall accuracy of the PINN approximation.

Visualization: 3D Scatter Plots at Terminal Time

- The terminal condition is $V(T, S_1, S_2) = \max(0.5S_1 + 0.5S_2 - 100, 0)$.
- We evaluate the PINN prediction on a grid in (S_1, S_2) at $t = T$.
- Three 3D scatter plots are produced:
  1. Exact terminal solution.
  2. PINN predicted terminal solution.
  3. Absolute error between prediction and exact.

Conclusions and Future Work You could consider

- PINNs provide a meshfree approach to solve high-dimensional PDEs.
- The two-asset Black–Scholes PDE for basket options can be solved by embedding the PDE and boundary conditions into a loss function.
- The presented approach is verified against the known terminal payoff.
- Future work may include:
    - More sophisticated network architectures.
    - Adaptive sampling strategies.
    - Application to more complex multi-asset or American options.